STL应用

2019年4月17日 16:42

1. 动态输入vector
   i. vector<int>v(n,0);//n是大小，0是初值
   ii. for(int &a : v)
   cin>>a;
2. 找序列中恰能加成m的两个数，且这两个数差值越大越好
   i. sort(begin,end)将[begin,end)范围按<顺序排序
   ii. find(begin,end,t)返回[begin,end)范围第一个值为t的位置，若找不到则返回end
   iii. sort(a,a+n);
   for__(i,0,n/2)
   if(**find(a,a+n,m-a[i])!=a+n**){
   printf("%d %d",a[i],m-a[i]);
   return 0;}
3. 从字符串s中截取前缀，要求该前缀子串的子集里可以找到足够字母数的字符串t
   for_(i,0,n)
   cnt[s[i]][++tms[s[i]]] = i+1;//求字母s[i]第tms次出现的位置
   for__(i,'a','z')
   ans= max(ans, cnt[i][**count(t.begin(),t.end(),i)**]);
4. 判断map中首元素是否含有a，若用if(m[a])会自动插入m[a]=0，应用m.count(a)
5. 判断110100100010000……序列第i个字是不是1
   i. 先打表各个1的编号
   ii. 再二分查找表里有没有这个编号，没有就说明是0
   iii. 利用*lower_bound(begin,end,t)为[begin,end)第一个>=t的数据，若它恰为t，说明t存在在这个表内。因为返回的是第一个，还不用管表的后半部分会不会有溢出导致恰为t（大概）
   iv. int idxof1[100000];　　//第i个1是第几位
   int main(){
   ios::sync_with_stdio(0);
   cin>>n;
   idxof1[0]= 1;
   for_(i,1,100000)
   idxof1[i]= idxof1[i-1] + i-1;
   for_(i,0,n){
   cin>>t;
   if(***lower_bound(idxof1,idxof1+100000,t) == t**)
   cout<<"1"<<endl;
   else
   cout<<"0"<<endl;}
6. 找二维坐标上x递增，y递减的点的数量，重叠时应重复计数
   i. pair<int,int>按首元判断大小，首元相等时按次元判断大小，利用这个性质，从小到大遍历时，可以保证x不减，而x不变时y不减，但不能保证x变化时y有什么关系
   ii. multiset允许将重叠元素视作两个元素
   iii. 注意：multiset删除部分元素后，迭代器位置会处于一个神秘的位置，可能会取值

为(0,0)，所以删元素很麻烦，我选择了不插入明显不符合的，和计数时不考虑不符合的

iv. 用p存储当前y最小的坐标对，利用multiset和二元组的排序法，看到x相同，则一定可以计，看到y小，也一定也可以计

v.
```
for_(i,0,n){
    cin>>x>>y;
    multiset< pair<int,int> >::iterator it= s.begin();
    bool ctn= 0;
    while(it!=s.end()){
        if(it->first < x && it->second <= y){
            ctn= 1;            //新输入的肯定不是，跳过
            break;}
        else if(it->first == x && it->second < y){
            ctn= 1;            //新输入的肯定不是，跳过
            break;}
        else if(it->first > x)
            break;
        ++it;}
    if(ctn)
        continue;
    s.insert(make_pair(x,y));}

int ans= 0;
multiset< pair<int,int> >::iterator it= s.begin();
pair <int,int> p= make_pair(0, 1<<30);
while(it!=s.end()){                    //x线性不减地遍历集合
    if(*it == p)
        ++ans;            //重叠
    else if(it->second < p.second)
        ++ans,            //更右的一列，且更偏下
        p= *it;
    ++it;}
cout<<ans;
```

7. 已知每层走到上一层的楼梯数，求a层走到b层的楼梯数

i.
```
if(a>b)
    swap(a,b);
```

ii.
```
cout<<accumulate(s+a,s+b,0ll);
```

8. 多元组tuple<元1类型，元2类型……>

i. 需要获得第x元时（下标从0开始），使用std::get<x>(元组数据引用)